Exploratory Data Analysis Wine Quality dataset

November 27, 2024

1 Exploratory Data Analysis Wine Quality dataset

Agenda

- Loading the dataset
- Data wrangling for missing variables.
- Data transformation.
- Data visualization
- Answering the main questions:

```
[ ]:
```

```
[ ]: import pandas as pd
```

```
[ ]: df_red = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/
      ↪wine-quality/winequality-red.csv", delimiter=";")
     df_white = pd.read_csv("https://archive.ics.uci.edu/ml/
      ↪machine-learning-databases/wine-quality/winequality-white.csv", delimiter=";
      ↪")
```

```
[115]: df_red.columns
```

```
[115]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
              'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
              'pH', 'sulphates', 'alcohol', 'quality'],
             dtype='object')
```

```
[116]: df_white.columns
```

```
[116]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
              'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
              'pH', 'sulphates', 'alcohol', 'quality'],
             dtype='object')
```

```
[117]: df_red.dtypes
```

```
[117]: fixed acidity           float64
       volatile acidity        float64
```

```
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

[118]:
```
df_red.iloc[100:110]
```

[118]:
```
     fixed acidity  volatile acidity  citric acid  …  sulphates  alcohol
quality
100            8.3             0.610         0.30  …       0.61     10.2
6
101            7.8             0.500         0.30  …       0.56     10.4
6
102            8.1             0.545         0.18  …       0.59      9.0
6
103            8.1             0.575         0.22  …       0.51      9.2
5
104            7.2             0.490         0.24  …       0.48      9.4
5
105            8.1             0.575         0.22  …       0.51      9.2
5
106            7.8             0.410         0.68  …       1.31      9.3
5
107            6.2             0.630         0.31  …       0.79      9.3
5
108            8.0             0.330         0.53  …       0.80      9.6
6
109            8.1             0.785         0.52  …       0.69      9.3
5

[10 rows x 12 columns]
```

[119]:
```
df_red.describe()
```

[119]:
```
       fixed acidity  volatile acidity  …      alcohol      quality
count    1599.000000       1599.000000  …  1599.000000  1599.000000
mean        8.319637          0.527821  …    10.422983     5.636023
std         1.741096          0.179060  …     1.065668     0.807569
min         4.600000          0.120000  …     8.400000     3.000000
25%         7.100000          0.390000  …     9.500000     5.000000
```

```
50%          7.900000           0.520000  …    10.200000      6.000000
75%          9.200000           0.640000  …    11.100000      6.000000
max         15.900000           1.580000  …    14.900000      8.000000

[8 rows x 12 columns]
```

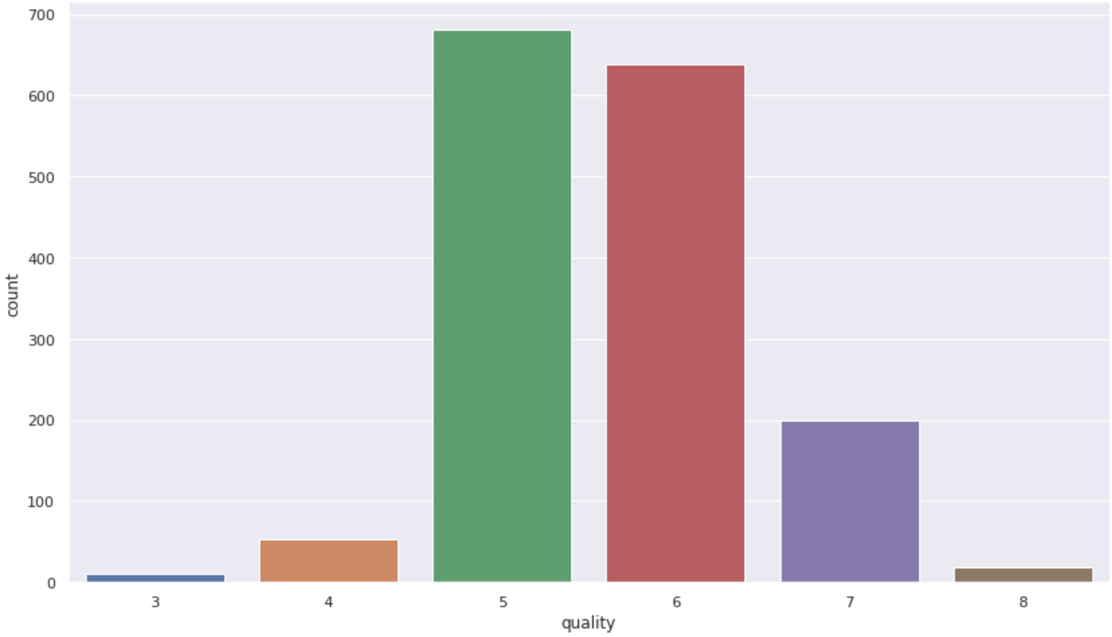
```
[120]: df_red.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1599 non-null float64
volatile acidity        1599 non-null float64
citric acid             1599 non-null float64
residual sugar          1599 non-null float64
chlorides               1599 non-null float64
free sulfur dioxide     1599 non-null float64
total sulfur dioxide    1599 non-null float64
density                 1599 non-null float64
pH                      1599 non-null float64
sulphates               1599 non-null float64
alcohol                 1599 non-null float64
quality                 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

2 Analysis of Red Wine

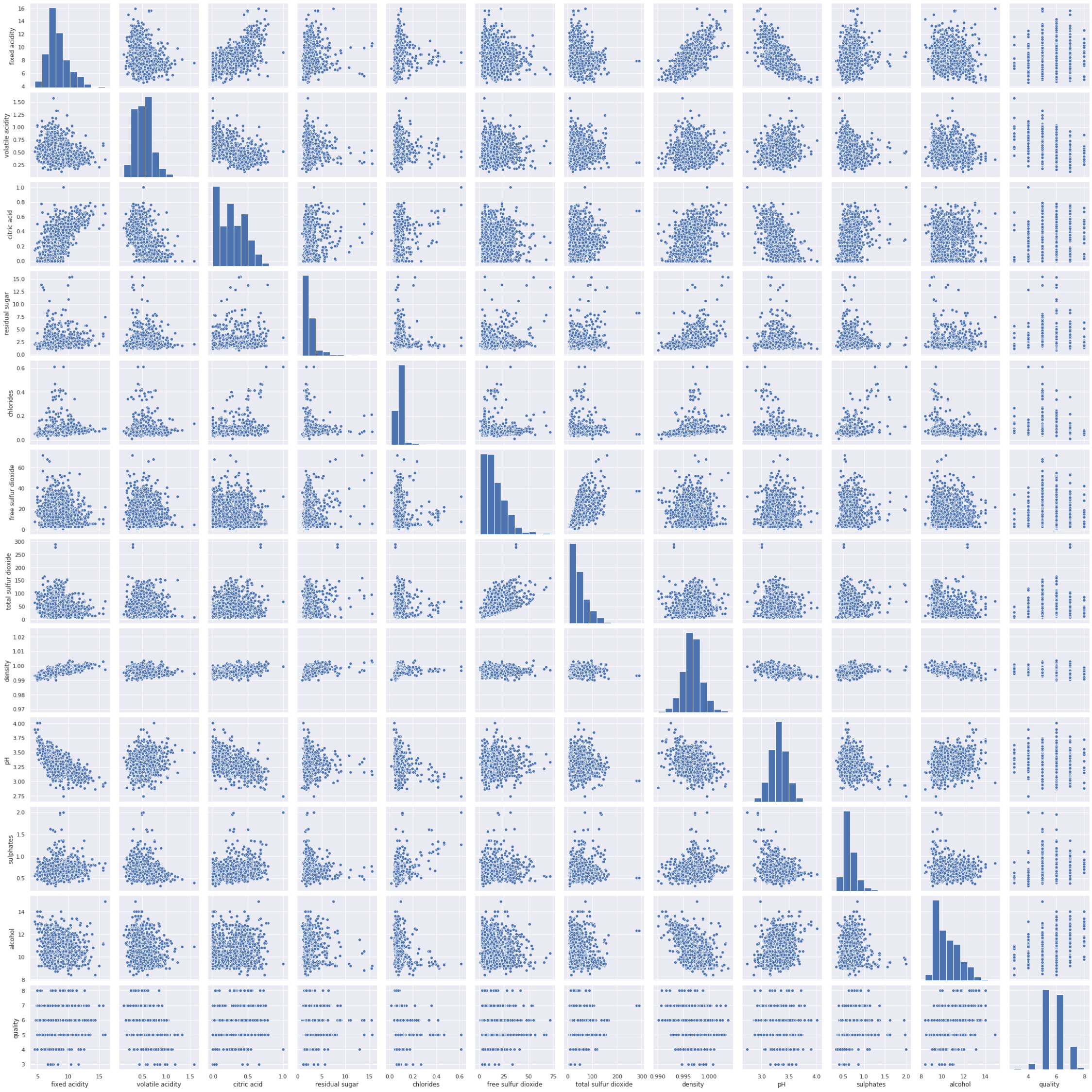
```
[121]: import seaborn as sns

       sns.set(rc={'figure.figsize': (14, 8)})
       sns.countplot(df_red['quality'])
```

```
[121]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7e56e31d0>
```



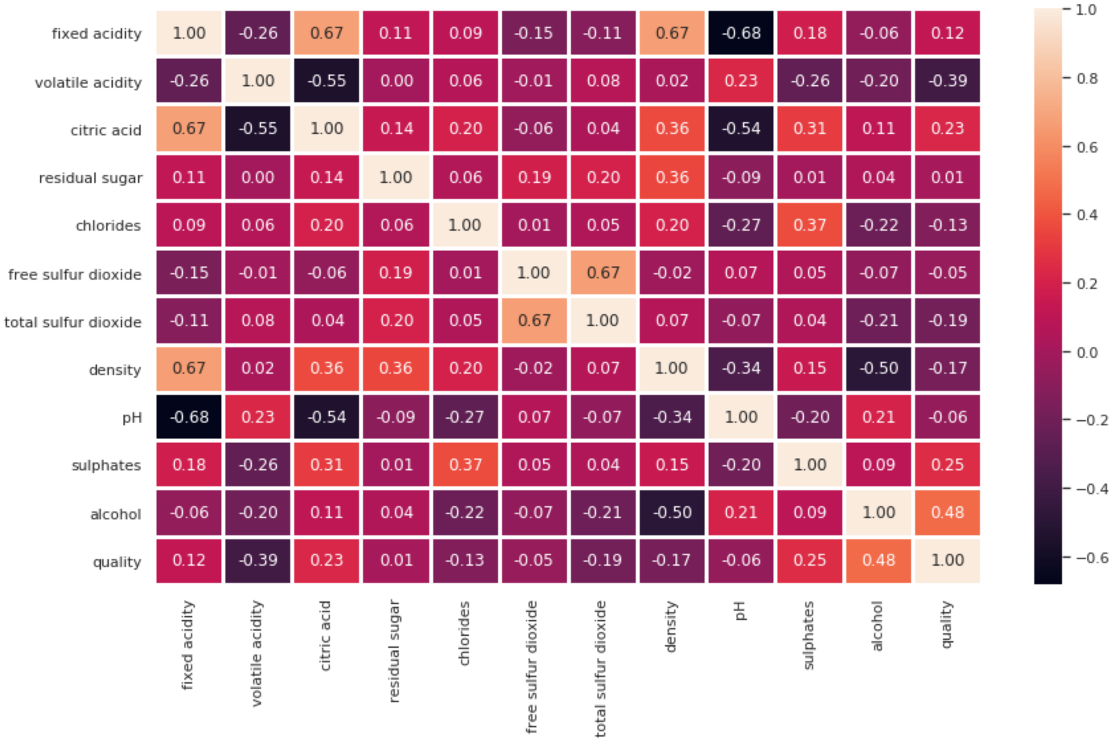
```
[122]: sns.pairplot(df_red)
```

```
[122]: <seaborn.axisgrid.PairGrid at 0x7fc7e56a0400>
```



```
[123]: sns.heatmap(df_red.corr(), annot=True, fmt='.2f', linewidths=2)
```

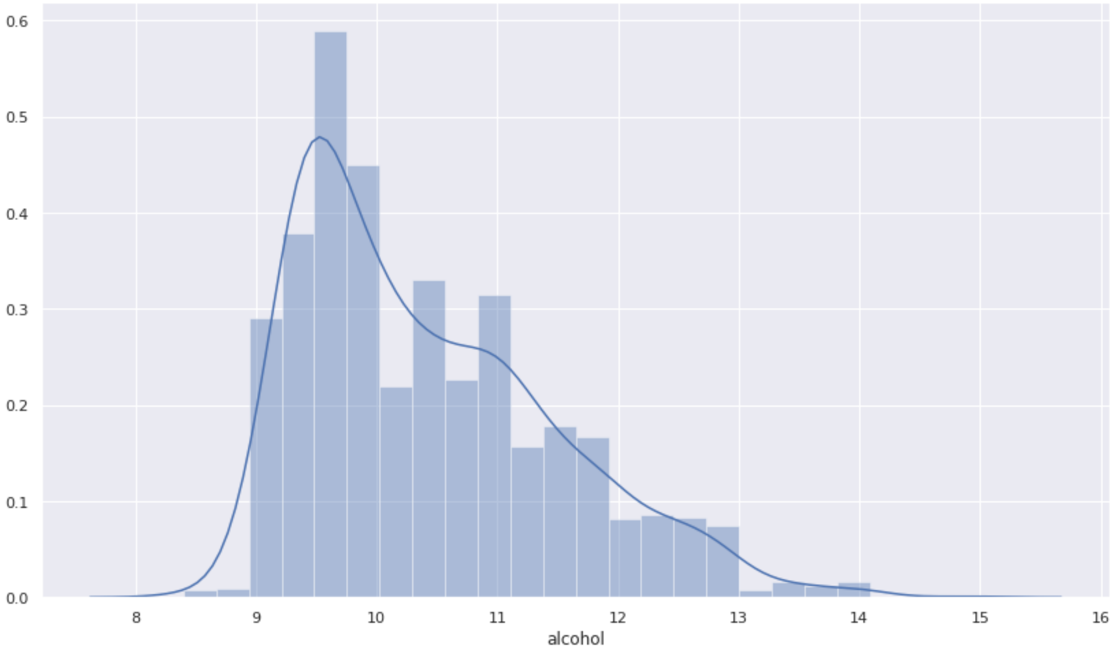
```
[123]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7e2f6b470>
```



```
[ ]:
```

```
[124]: sns.distplot(df_red['alcohol'])
```

```
[124]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7e2549668>
```

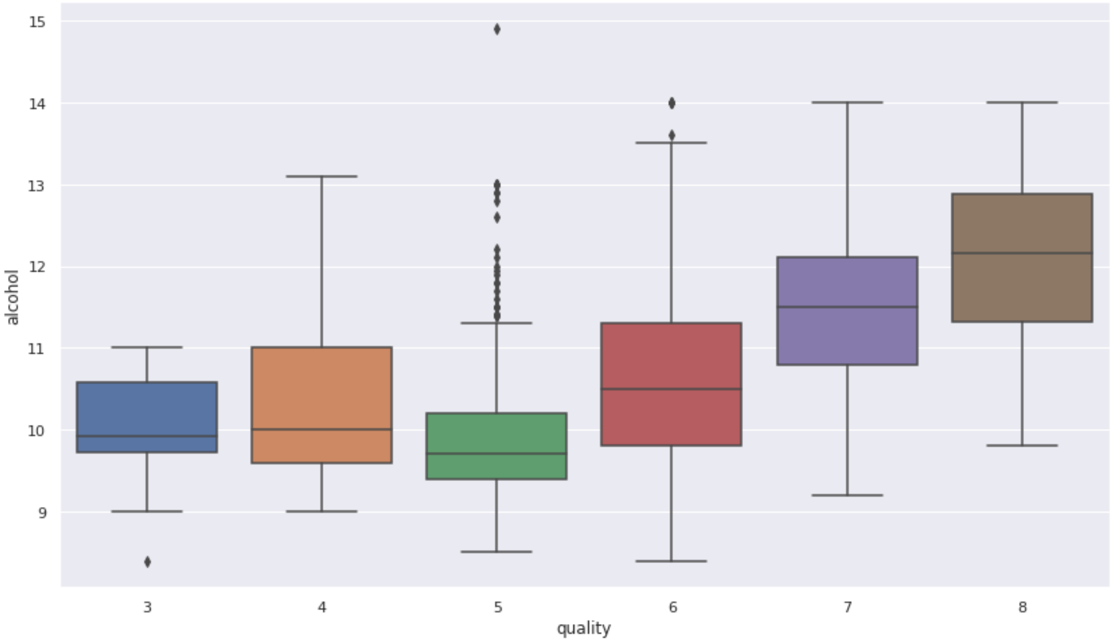


```
[125]: from scipy.stats import skew

       skew(df_red['alcohol'])
```

```
[125]: 0.8600210646566755
```

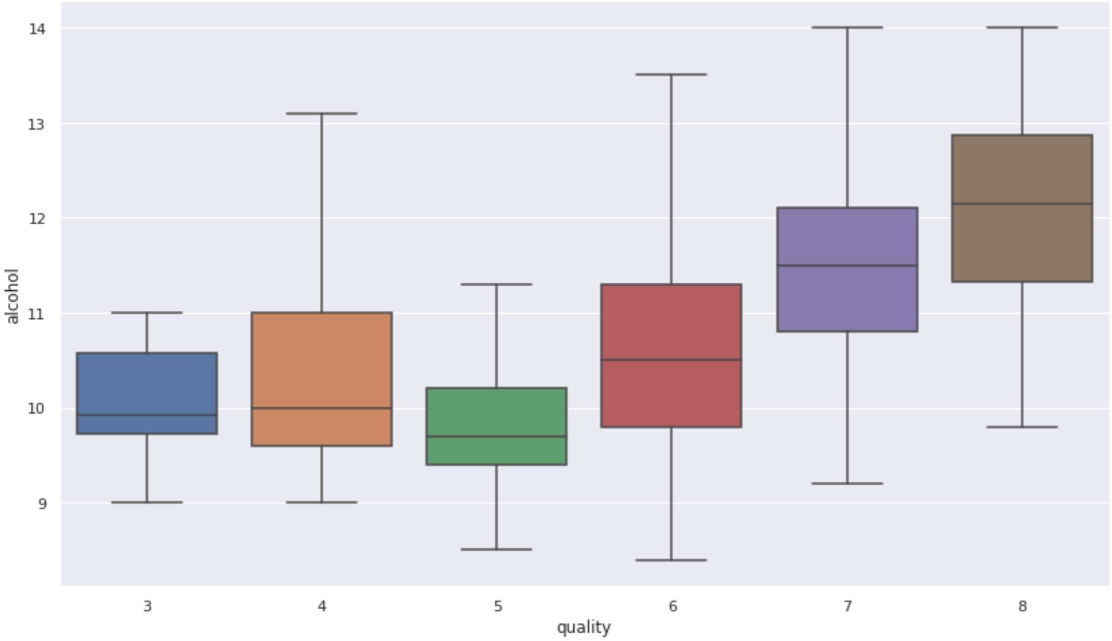
```
[126]: sns.boxplot(x='quality', y='alcohol', data = df_red)
```

```
[126]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7e2417940>
```



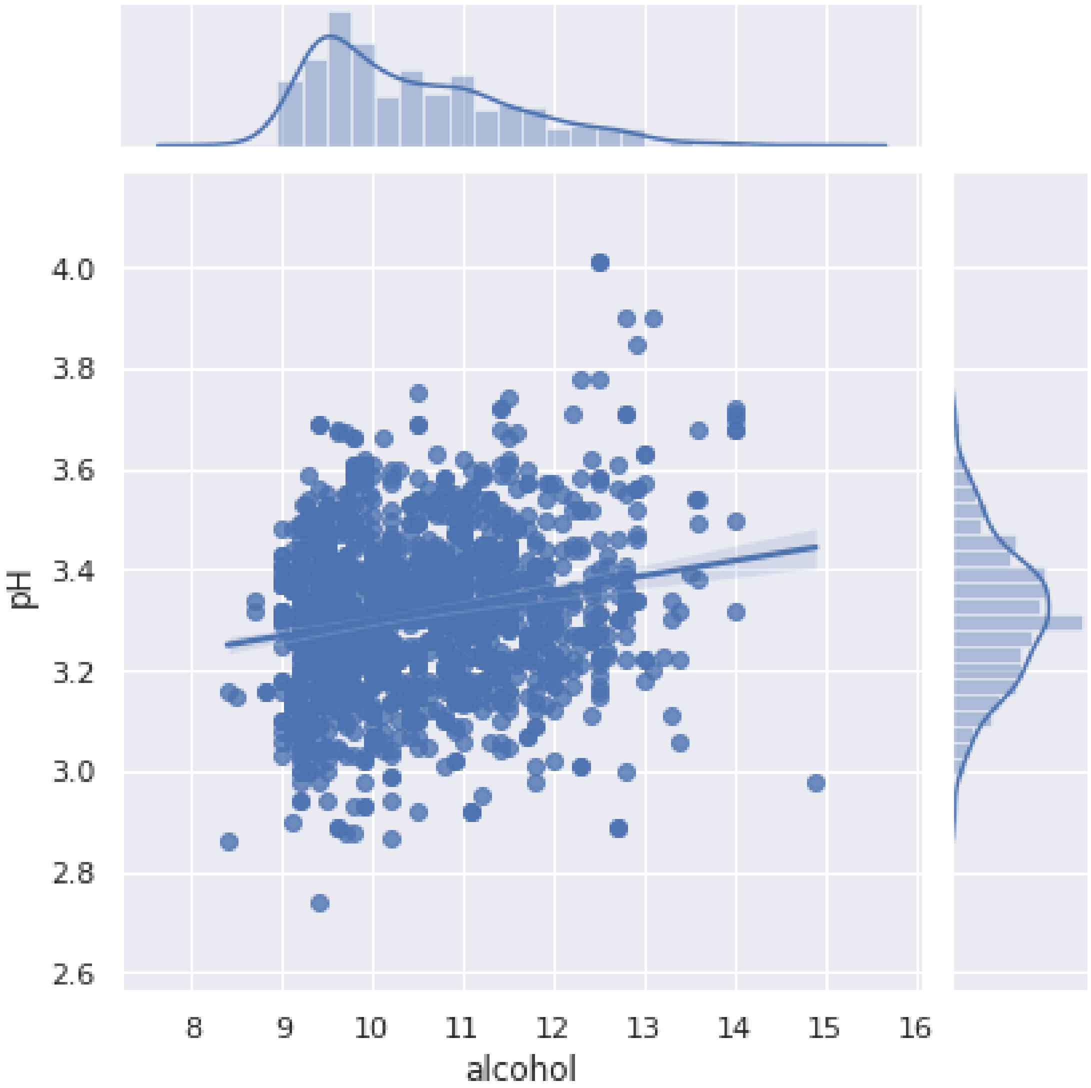
```
[127]: sns.boxplot(x='quality', y='alcohol', data = df_red, showfliers=False)
```

```
[127]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7e23be748>
```



```
[128]: sns.jointplot(x='alcohol',y='pH',data=df_red, kind='reg')
```

```
[128]: <seaborn.axisgrid.JointGrid at 0x7fc7e246e470>
```



```
[ ]: from scipy.stats import pearsonr

     def get_correlation(column1, column2, df):
       pearson_corr, p_value = pearsonr(df[column1], df[column2])
       print("Correlation between {} and {} is {}".format(column1, column2,␣
      ↪pearson_corr))
       print("P-value of this correlation is {}".format(p_value))
```

```
[130]: get_correlation('alcohol','pH', df_red)
```

```
Correlation between alcohol and pH is 0.20563250850549825
```

```
P-value of this correlation is 9.964497741465556e-17
```

[131]:
```
df_white.describe()
```

[131]:
```
       fixed acidity  volatile acidity  …      alcohol      quality
count    4898.000000       4898.000000  …  4898.000000  4898.000000
mean        6.854788          0.278241  …    10.514267     5.877909
std         0.843868          0.100795  …     1.230621     0.885639
min         3.800000          0.080000  …     8.000000     3.000000
25%         6.300000          0.210000  …     9.500000     5.000000
50%         6.800000          0.260000  …    10.400000     6.000000
75%         7.300000          0.320000  …    11.400000     6.000000
max        14.200000          1.100000  …    14.200000     9.000000

[8 rows x 12 columns]
```

3 White wine analysis

[153]:
```
print("white mean = ",df_white["quality"].mean())
print("red mean =",df_red["quality"].mean())
```

```
white mean =  5.87790935075541
red mean = 5.6360225140712945
```

[154]:
```
d = {'color': ['red','white'], 'mean_quality': [5.636023,5.877909]}
df_mean = pd.DataFrame(data=d)
df_mean
```

[154]:
```
   color  mean_quality
0    red      5.636023
1  white      5.877909
```

[ ]:
```
# Let us add new attribute called wine_category  to the both dataframe

df_white['wine_category'] = 'white'
df_red['wine_category'] = 'red'
```

[158]:
```
print('RED WINE: List of "quality"', sorted(df_red['quality'].unique()))
print('WHITE WINE: List of "quality"', sorted(df_white['quality'].unique()))
```

```
RED WINE: List of "quality" [3, 4, 5, 6, 7, 8]
WHITE WINE: List of "quality" [3, 4, 5, 6, 7, 8, 9]
```

4 Convert into categorical dataset

[ ]:

```
df_red['quality_label'] = df_red['quality'].apply(lambda value: ('low' if value␣
 ↪<= 5 else 'medium') if value <= 7 else 'high')
df_red['quality_label'] = pd.Categorical(df_red['quality_label'],␣
 ↪categories=['low', 'medium', 'high'])

df_white['quality_label'] = df_white['quality'].apply(lambda value: ('low' if␣
 ↪value <= 5 else 'medium') if value <= 7 else 'high')
df_white['quality_label'] = pd.Categorical(df_white['quality_label'],␣
 ↪categories=['low', 'medium', 'high'])
```

[163]:

```
print(df_white['quality_label'].value_counts())
df_red['quality_label'].value_counts()
```

```
medium    3078
low       1640
high       180
Name: quality_label, dtype: int64
```

[163]:

```
medium    837
low       744
high       18
Name: quality_label, dtype: int64
```

[170]:

```
df_wines = pd.concat([df_red, df_white])

# Re-shuffle records just to randomize data points.
# `drop=True`: this resets the index to the default integer index.
df_wines = df_wines.sample(frac=1.0, random_state=42).reset_index(drop=True)
df_wines.head(10)
```

[170]:

```
   fixed acidity  volatile acidity  …  wine_category  quality_label
0            7.0              0.17  …          white           high
1            7.7              0.64  …            red            low
2            6.8              0.39  …          white         medium
3            6.3              0.28  …          white         medium
4            7.4              0.35  …          white         medium
5            7.2              0.53  …            red         medium
6            7.5              0.27  …          white            low
7            6.8              0.11  …          white         medium
8            9.0              0.44  …            red            low
9            7.1              0.23  …          white         medium

[10 rows x 14 columns]
```

[168]:

```
subset_attr = ['alcohol', 'density', 'pH', 'quality']

low = round(df_wines[df_wines['quality_label'] == 'low'][subset_attr].
 ↪describe(), 2)
medium = round(df_wines[df_wines['quality_label'] == 'medium'][subset_attr].
 ↪describe(), 2)
high = round(df_wines[df_wines['quality_label'] == 'high'][subset_attr].
 ↪describe(), 2)

pd.concat([low, medium, high], axis=1,
          keys=['  Low Quality Wine',
                '  Medium Quality Wine',
                '  High Quality Wine'])
```

[168]:

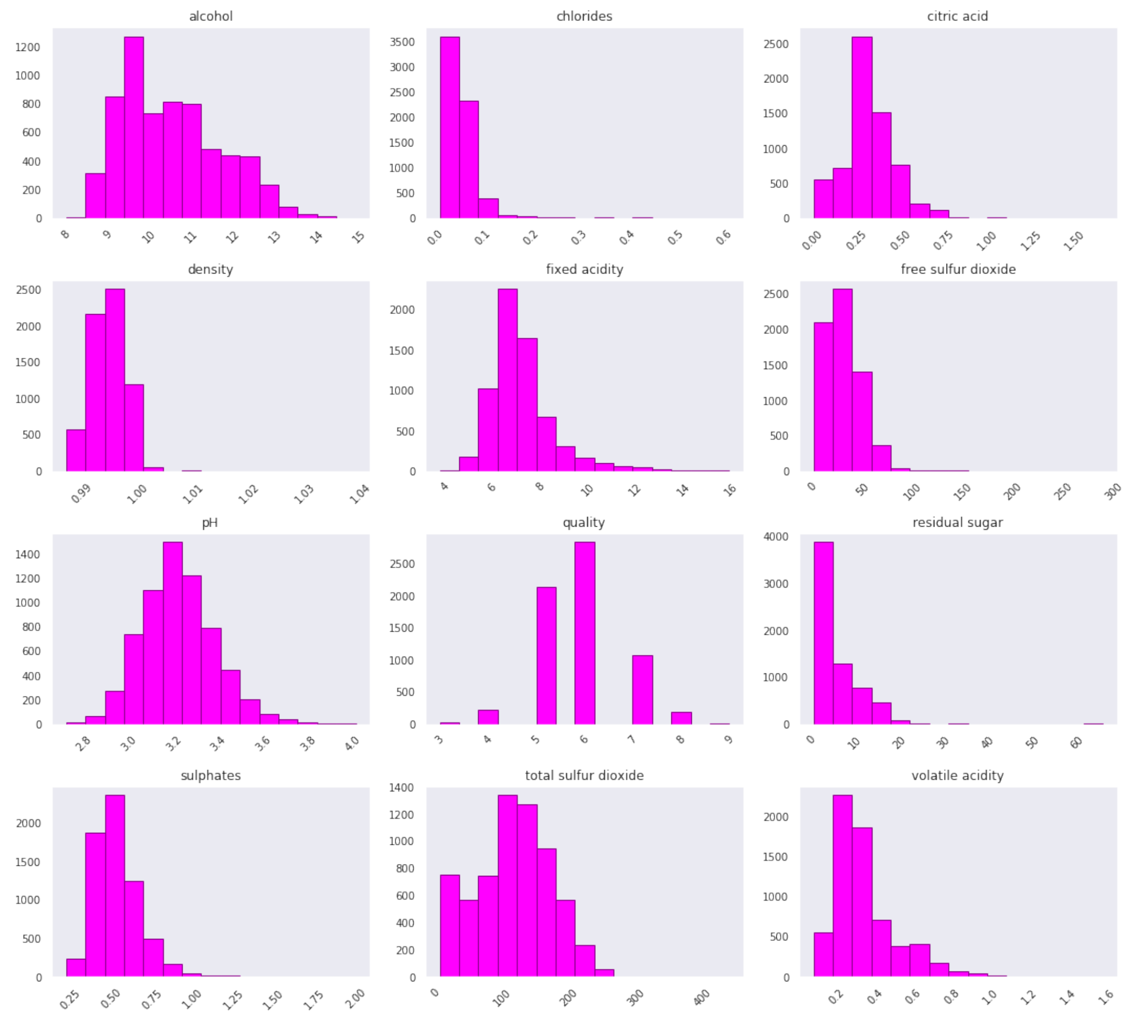
```
       Low Quality Wine                    …  High Quality Wine
                alcohol  density       pH  …            density       pH
quality
count           2384.00  2384.00  2384.00  …             198.00   198.00
198.00
mean               9.87     1.00     3.21  …               0.99     3.23
8.03
std                0.84     0.00     0.16  …               0.00     0.16
0.16
min                8.00     0.99     2.74  …               0.99     2.88
8.00
25%                9.30     0.99     3.11  …               0.99     3.13
8.00
50%                9.60     1.00     3.20  …               0.99     3.23
8.00
75%               10.40     1.00     3.31  …               0.99     3.33
8.00
max               14.90     1.00     3.90  …               1.00     3.72
9.00

[8 rows x 12 columns]
```

[ ]:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

[176]:

```
fig = df_wines.hist(bins=15, color='fuchsia', edgecolor='darkmagenta',␣
 ↪linewidth=1.0, xlabelsize=10, ylabelsize=10, xrot=45, yrot=0,␣
 ↪figsize=(10,9), grid=False)

plt.tight_layout(rect=(0, 0, 1.5, 1.5))
```

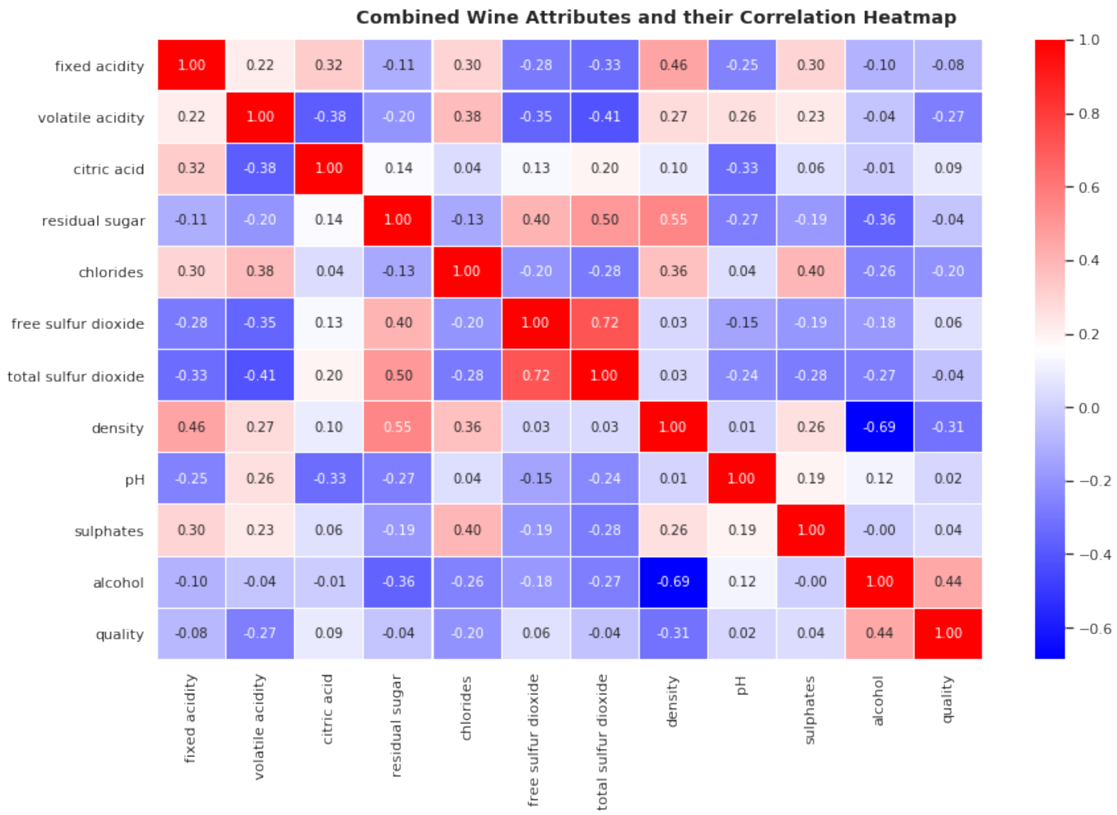


[160]:
```
df_red.head()
```

[160]:
```
   fixed acidity  volatile acidity  …  wine_category  quality_label
0            7.4              0.70  …            red            low
1            7.8              0.88  …            red            low
2            7.8              0.76  …            red            low
3           11.2              0.28  …            red         medium
4            7.4              0.70  …            red            low

[5 rows x 14 columns]
```

[185]:
```
fig, (ax) = plt.subplots(1, 1, figsize=(14,8))

hm = sns.heatmap(df_wines.corr(),
                 ax=ax,
                 cmap="bwr",
                 annot=True,
                 fmt='.2f',
                 linewidths=.05)

fig.subplots_adjust(top=0.94)
fig.suptitle('Combined Wine Attributes and their Correlation Heatmap',
              fontsize=14,
              fontweight='bold')
```

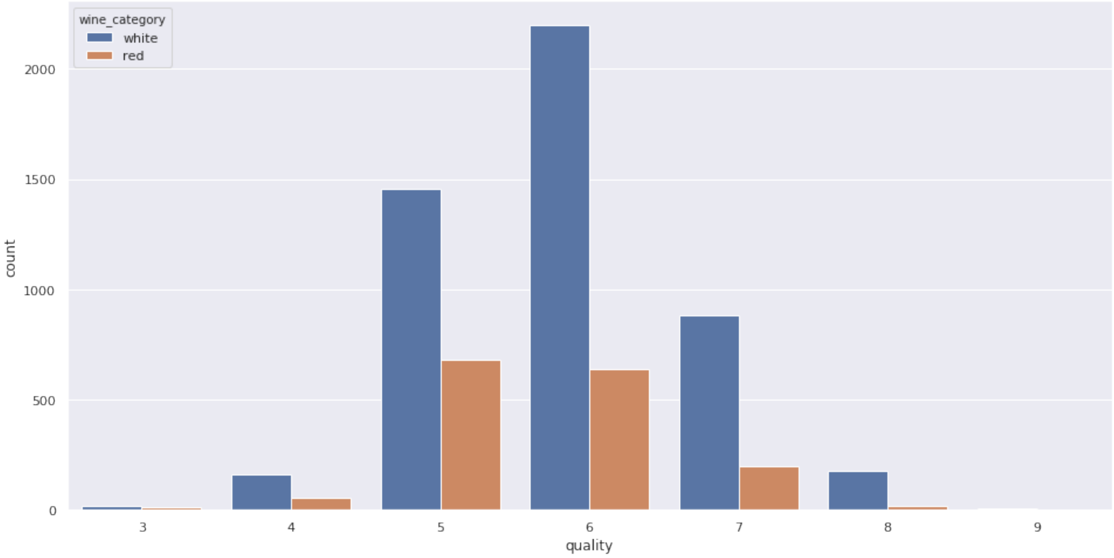
```
[185]: Text(0.5, 0.98, 'Combined Wine Attributes and their Correlation Heatmap')
```



4.1 Discrete categorical attributes

```
[188]: fig = plt.figure(figsize=(16, 8))

       sns.countplot(data=df_wines, x="quality", hue="wine_category")
```

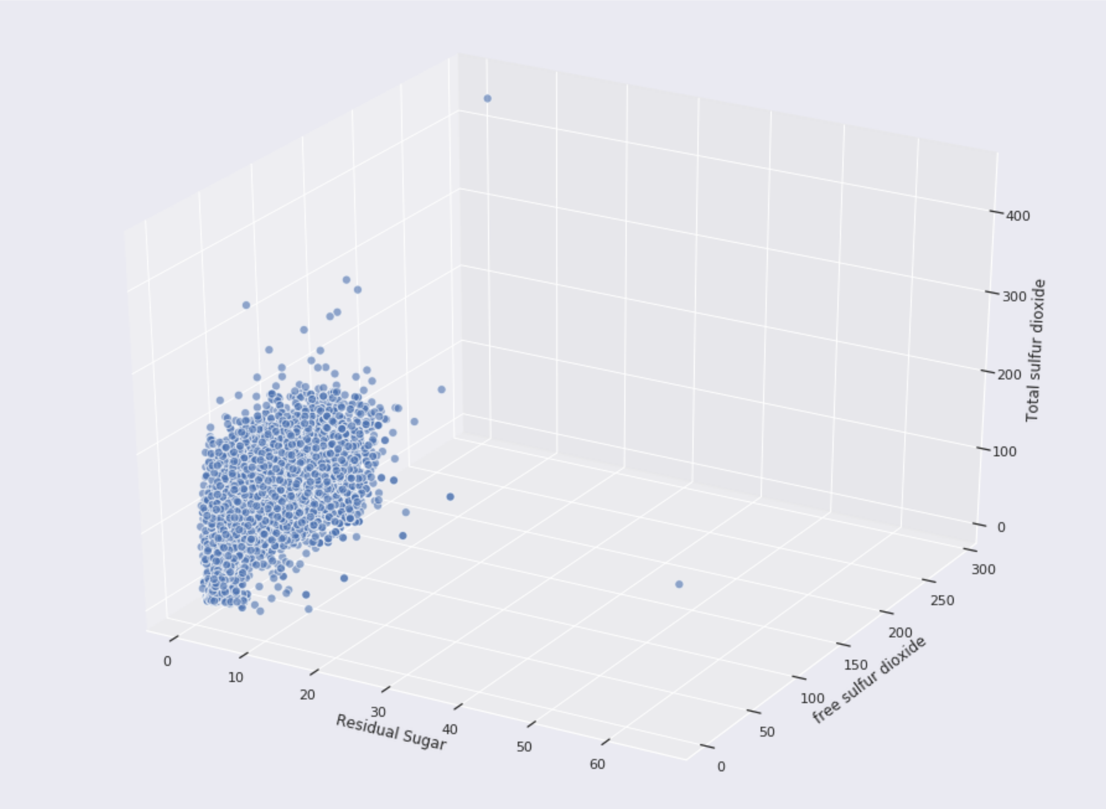
```
[188]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7e196da20>
```



4.2 3D visualization

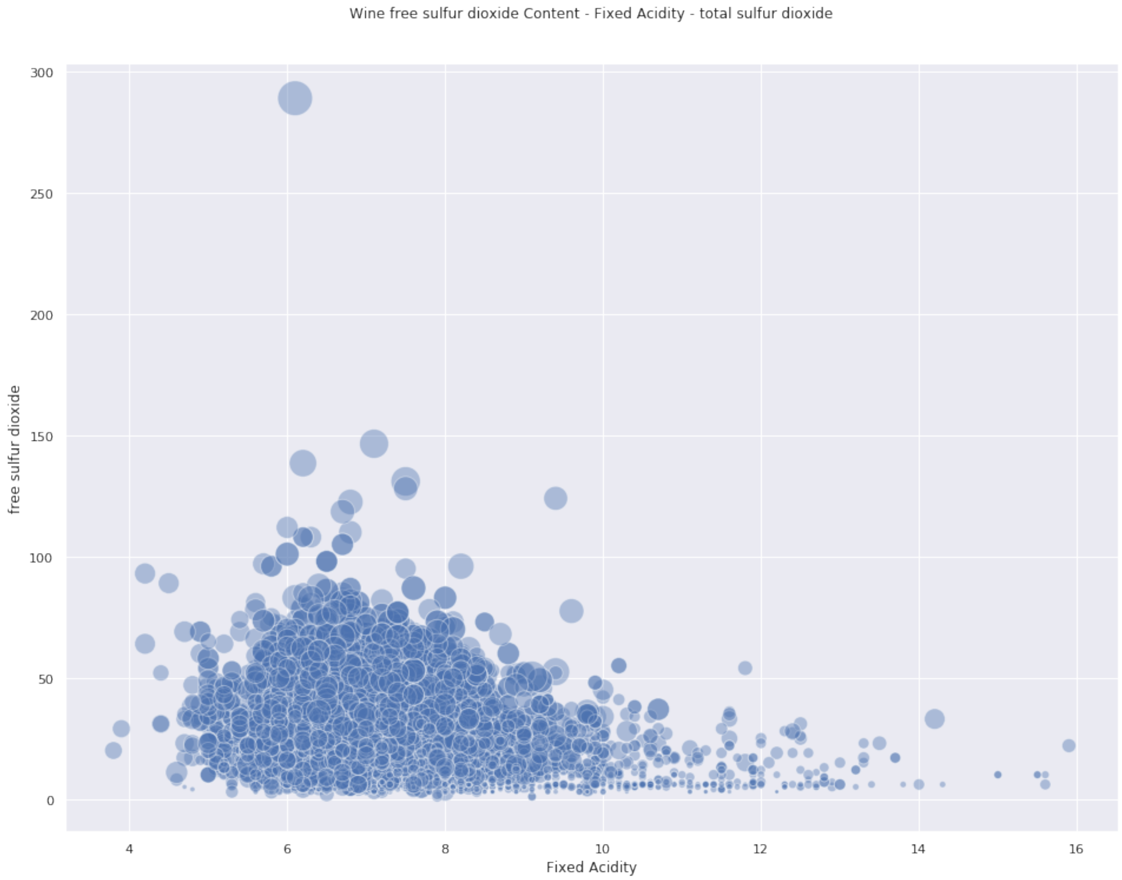
[196]:

```
fig = plt.figure(figsize=(16, 12))
ax = fig.add_subplot(111, projection='3d')

xscale = df_wines['residual sugar']
yscale = df_wines['free sulfur dioxide']
zscale = df_wines['total sulfur dioxide']
ax.scatter(xscale, yscale, zscale, s=50, alpha=0.6, edgecolors='w')

ax.set_xlabel('Residual Sugar')
ax.set_ylabel('free sulfur dioxide')
ax.set_zlabel('Total sulfur dioxide')

plt.show()
```



```
[200]: fig = plt.figure(figsize=(16, 12))

       plt.scatter(x = df_wines['fixed acidity'],
                   y = df_wines['free sulfur dioxide'],
                   s = df_wines['total sulfur dioxide'] * 2,
                   alpha=0.4,
                   edgecolors='w')

       plt.xlabel('Fixed Acidity')
       plt.ylabel('free sulfur dioxide')
       plt.title('Wine free sulfur dioxide Content - Fixed Acidity - total sulfur␣
        ↪dioxide', y=1.05)
```

```
[200]: Text(0.5, 1.05, 'Wine free sulfur dioxide Content - Fixed Acidity - total sulfur
       dioxide')
```



[195]:
```
df_wines.columns
```

[195]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality', 'wine_category',
       'quality_label'],
      dtype='object')
```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC,SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier,AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

[ ]:
```
from sklearn.model_selection import train_test_split,cross_validate
from sklearn.preprocessing import MinMaxScaler,StandardScaler,LabelEncoder
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
```

```
[ ]: label_quality = LabelEncoder()

     df_wines['quality_label'] = label_quality.
      ↪fit_transform(df_wines['quality_label'])
```

```
[236]: df_wines.tail(10)
```

```
[236]:       fixed acidity  volatile acidity  …  wine_category  quality_label
       6487            6.1              0.22  …              1              2
       6488           10.3              0.50  …              0              2
       6489            6.4              0.31  …              1              1
       6490            5.9              0.26  …              1              1
       6491            8.0              0.34  …              1              0
       6492            7.6              0.32  …              1              1
       6493            5.6              0.28  …              1              2
       6494            6.4              0.37  …              1              1
       6495            6.5              0.26  …              1              1
       6496            7.2              0.62  …              0              1

       [10 rows x 14 columns]
```

```
[ ]: x_train,x_test,y_train,y_test=train_test_split(df_wines.
      ↪drop(['quality','wine_category'],axis=1),df_wines['quality_label'],test_size=0.
      ↪30,random_state=42)

     models=[LogisticRegression(),
             LinearSVC(),
             SVC(kernel='rbf'),
             KNeighborsClassifier(),
             RandomForestClassifier(),
             DecisionTreeClassifier(),
             GradientBoostingClassifier(),
             GaussianNB()]
```

```
[242]: model_names=['LogisticRegression',
                    'LinearSVM',
                    'rbfSVM',
                    'KNearestNeighbors',
                    'RandomForestClassifier',
                    'DecisionTree',
                    'GradientBoostingClassifier',
                    'GaussianNB']

       acc=[]
       eval_acc={}

       for model in range(len(models)):
```

```
    classification_model=models[model]
    classification_model.fit(x_train,y_train)
    pred=classification_model.predict(x_test)
    acc.append(accuracy_score(pred,y_test))

eval_acc={'Modelling Algorithm':model_names,'Accuracy':acc}
eval_acc
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[242]:

```
{'Accuracy': [0.9687179487179487,
  0.9733333333333334,
  0.6051282051282051,
  0.6912820512820513,
  1.0,
  1.0,
  1.0,
  1.0],
 'Modelling Algorithm': ['LogisticRegression',
  'LinearSVM',
  'rbfSVM',
  'KNearestNeighbors',
  'RandomForestClassifier',
  'DecisionTree',
  'GradientBoostingClassifier',
  'GaussianNB']}
```

[239]:

```
acc_table=pd.DataFrame(eval_acc)
acc_table = acc_table.sort_values(by='Accuracy', ascending=[False])
acc_table
```

[239]:

```
         Modelling Algorithm  Accuracy
4     RandomForestClassifier  1.000000
5               DecisionTree  1.000000
```

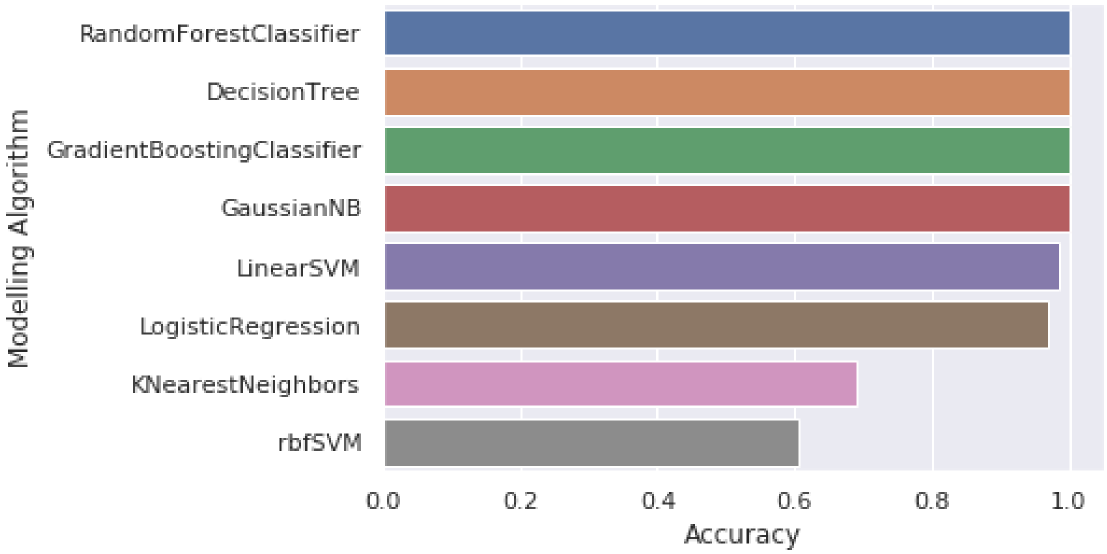
```
6  GradientBoostingClassifier  1.000000
7                  GaussianNB  1.000000
1                   LinearSVM  0.983590
0          LogisticRegression  0.968718
3           KNearestNeighbors  0.691282
2                      rbfSVM  0.605128
```

[240]:

```
sns.barplot(y='Modelling Algorithm',x='Accuracy',data=acc_table)
```

[240]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc7e0624c50>
```



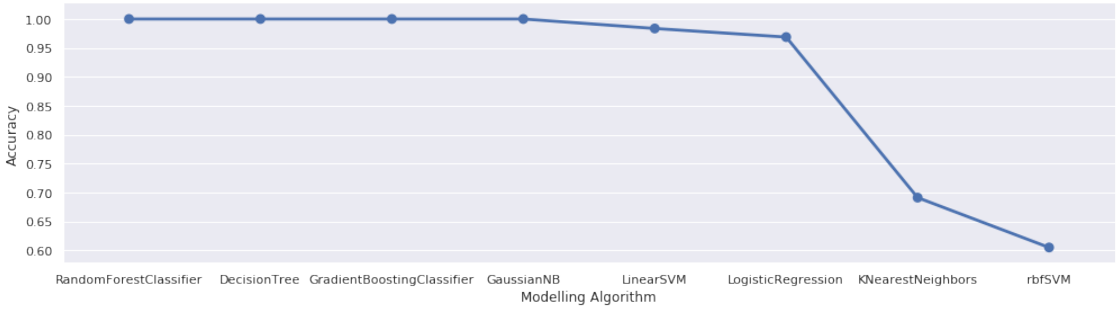
[241]:

```
sns.catplot(x='Modelling Algorithm',y='Accuracy',data=acc_table,kind='point',size=4,aspect=3.5)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:3695: UserWarning:
The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

[241]:

```
<seaborn.axisgrid.FacetGrid at 0x7fc7e0efab38>
```


Accuracy
RandomForestClassifier
DecisionTree
GradientBoostingClassifier
GaussianNB
LinearSVM
LogisticRegression
KNearestNeighbors
rbfSVM
Modelling Algorithm